UMCG Synthea patient generator webservice

Joshua Rubingh

Created on : November, 2020

Last updated : November, 2020

Table of contents

List of Figures

List of Tables

Here you can read more information about the UMCG Synthea patient generator webservice.

This webservice uses Django for the front-end and Synthea for the patient generating.

Chapter 1

Installation

In order to install this Virtual Research Environment project, we use the following packages / software.

- Synthea
- NGINX
- Django

First we need to checkout the code.

```
git clone https://git.web.rug.nl/P300021/synthea_webservice /opt/deploy/synthea_webservice
```

1.1 Synthea

TODO: Make clear setup for synthea. As we are using NL demographical data.

1.2 NGINX

Install NGINX through the package manager. For Ubuntu this would be

```
sudo apt install nginx
```

Also configure SSL to make the connections secure. This is outside this installation scope.

1.2.1 Setup

After installation of the packages, create a symbolic link in the */etc/nginx/sites-enabled* so that a new VHost is created.

Important parts of the VHost configuration:

```
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
```

(continues on next page)

(continued from previous page)

```
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#

server {
    listen 80;
    listen [::]:80;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name localhost;

    access_log /var/log/nginx/synthea_webservice.access.log;
    error_log /var/log/nginx/synthea_webservice.error.log;

    location /static {
        alias /opt/deploy/synthea_webservice/webservice/static;
    }

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        # try_files $uri $uri/ =404;

        proxy_pass         http://localhost:8000;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection "upgrade";

        proxy_redirect     off;
        proxy_set_header   Host $host;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Host $sent_http_host;
        proxy_set_header   X-Forwarded-Proto $scheme;
```

(continues on next page)

(continued from previous page)

```
    }
}
```

In order to test if NGINX is configured correctly run *nginx -t* and it should give an OK message:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

1.3 Django

We install Django with standard settings. We could run it in Aync way, but then you need some more steps: https://docs.djangoproject.com/en/3.0/howto/deployment/asgi/ So for now, we keep it simple.

Create a python virtual environment

```
cd /opt/deploy/synthea_webservice
python3 -m venv venv
source venv/bin/activate
```

Finally we install the required Python modules

```
pip install -r requirements
```

This will install all the needed Python modules we need to run this Django project.

1.3.1 Settings

The settings for Django are set in an *.env* file so that you can easily change the environment from production to testing. There is an *.env.example* file that could be used as a template.

```
# A uniquely secret key
SECRET_KEY=@wb=#(f4vc0l(e!5*eo+a@flnxb2@!l9!=c6w=4b+x$=!8&vy%'

# Disable debug in production
DEBUG=True

# Allowed hosts that Django does server. Take care when NGINX is proxying infront of Django
ALLOWED_HOSTS=127.0.0.1,localhost

# All internal IPS for Django. Use comma separated list
INTERNAL_IPS=127.0.0.1

# Enter the database url connection: https://github.com/jacobian/dj-database-url
DATABASE_URL=sqlite:////opt/deploy/synthea_webservice/webservice/db.sqlite3

# The location on disk where the static files will be placed during deployment. Setting is required
STATIC_ROOT=

# Enter the default timezone for the visitors when it is not known.
TIME_ZONE=Europe/Amsterdam

# Email settings

# Mail host
EMAIL_HOST=

# Email user name
```

(continues on next page)

(continued from previous page)

```
EMAIL_HOST_USER=na

# Email password
EMAIL_HOST_PASSWORD=na

# Email server port number to use
EMAIL_PORT=25

# Does the email server supports TLS?
EMAIL_USE_TLS=yes

# The sender address. This needs to be one of the allowed domains due to SPF checks
# The code will use a reply-to header to make sure that replies goes to the researcher and not this␣
↪address
EMAIL_FROM_ADDRESS=Do not reply<no-reply@rug.nl>

# The base folder where Synthea is installed. This folder should contain the file 'run_synthea.(|bat)'
settings.SYNTHEA_BASE_DIR = settings.BASE_DIR / '../synthea'

# The base output folder where Synthea will generate its output. This location will be appended with a␣
↪unique folder name.
settings.SYNTHEA_OUTPUT_DIR = settings.BASE_DIR / '../synthea_output'

# The location where the Synthea modules are located.
settings.SYNTHEA_MODULE_DIR = settings.SYNTHEA_BASE_DIR / 'src/main/resources/modules/'

# The location where the Synthea resources are located. This should also include the geography data.
settings.SYNTHEA_RESOURCE_DIR = settings.SYNTHEA_BASE_DIR / 'src/main/resources/'

# The output type for Synthea.
settings.SYNTHEA_EXPORT_TYPE = 'fhir_stu3'
```

Next we have to make the database structure. If you are using SQLite3 as a backend, make sure the database file **DOES** exist on disk.

```
touch /opt/deploy/synthea_webservice/webservice/db.sqlite3
```

Then in the Python virtual environment we run the following commands:

```
./manage.py migrate
./manage.py createsuperuser
./manage.py collectstatic
```

And finally you should be able to start the Django application

```
./manage.py runserver
```

Chapter 2

Models

2.1 Base

class lib.models.base.MetaDataModel(*args, **kwargs)

This is an abstract Django model with some general meta fields that can be used for other models.

created_at

The date and time when the model has been created. This will be automatically set once during creating.

Type datetime

updated_at

The date and time when the model has been updated. This will be automatically updated when the model is updated.

Type datetime

2.2 API

class apps.api.models.Token(*args, **kwargs)

Token model that holds all the tokens that are used for the API authentication.

A new token is generated every time when a new user is created. So there is no need for manual token creating. This is done through a signal *create_user_token*

user

The user to which this token belongs too

Type User

key

The key value that is used for token lookups

Type str

secret

The secret that is used for encrypting/signing the API messages

Type str

last_access

The date and time when the token is last used (logged in)

Type datetime

exception DoesNotExist

exception MultipleObjectsReturned

`is_supertoken()`
Boolean check if the token is belonging to a user with super user rights. Then this token is a super token.

Returns Returns true when the token belongs to a super user.
Return type Boolean

`class apps.api.models.TokenManager(*args, **kwargs)`
Custom queryset which will prefetch related user table data when requesting a token from the database as the user is mostly needed every time the token is requested.

`get_queryset()`
Return a new QuerySet object. Subclasses can override this method to customize the behavior of the Manager.

2.3 Synthea

`class apps.synthea.models.Synthea(*args, **kwargs)`
Synthea model that holds some iformation about a generated output.

`id`
A unique ID for every Synthea run. Leave empty for auto generating a new value.

Type uuid

`state`
The state for which you want to generate Synthea patient data

Type str

`population`
The amount of patients you want to generate

Type int

`gender`
Generate only male (m), only female(f) or leave empty for male and female

Type datetime

`age`
The age range for the patients. Enter like [min_age]-[max_age]

Type str

`module`
The module to use for patient generating. Leave empty for random use by Synthea

Type str

`log`
The outcome of a single patient generating run. This can be read in the admin area.

Type str

Returns A new Synthea model
Return type *Synthea*

`exception DoesNotExist`

`exception MultipleObjectsReturned`

`generate()`
Run the patient generation. This will return a logfile and a zipfile location for download.

The log will be stored in the model when done. This log can then be seen/readed in the admin section of Django

Returns The zip file location on disk.
Return type str

Chapter 3

Views

`class apps.api.views.Info(**kwargs)`

Show some API information. Also this can be used to check if the Hawk credentials are working.

Make sure your request does contain the header 'Content-Type': 'application/json'

`get(request, format=None)`

Default API get action will return the following information in a dict:

- Connected user
- Used authentication scheme
- The remote IP of the connection
- The used content type
- The full url to the API documentation (OpenAPI)
- If a super token is used

Chapter 4

API

Chapter 5

Utils

5.1 General

lib.utils.general.generate_encryption_key(*length=32*)
Generate a new encryption key of *length* chars. This is done by using the *get_random_string()* function with a default of 32 chars.

Parameters `length` (*int , optional*) – The length in chars of the encryption key. Defaults to 32.
Returns A string of *length* chars.
Return type str

lib.utils.general.get_ip_address(*request*)
Get the IP address of the requesting viewer. This is done by looking into the following variables in the headers.

1. HTTP_X_FORWARDED_FOR
2. REMOTE_ADDR

Parameters `request` (*BaseRequest*) – The Django request.
Returns IP address of the request
Return type str

lib.utils.general.get_random_int_value(*length=6*)
Generate a random number of *length* length numbers.

Parameters `length` (*int , optional*) – The length of the random number in amount of numbers. Defaults to 6.
Returns Returns a random number of 'length' numbers.
Return type int

lib.utils.general.get_random_string(*length=8*)
Generate a random string of *length* length characters.

Parameters `length` (*int , optional*) – The length of the random string in amount of characters. Defaults to 8.
Returns Returns a random string of 'length' characters.
Return type str

lib.utils.general.remove_html_tags(*text*)
Remove HTML tags and code from the input text

Parameters `text` (*str*) – Input text to be cleaned from HTML
Returns Cleaned HTML.
Return type str

class lib.utils.emails.EmailMultiRelated(*subject=''*, *body=''*, *from_email=None*, *to=None*, *bcc=None*, *connection=None*, *attachments=None*, *headers=None*, *alternatives=None*)

A version of EmailMessage that makes it easy to send multipart/related messages. For example, including text and HTML versions with inline images.

Returns EmailMultiAlternatives class
Return type EmailMultiAlternatives

attach_related(*filename=None*, *content=None*, *mimetype=None*)

Attaches a file with the given filename and content. The filename can be omitted and the mimetype is guessed, if not provided.

If the first parameter is a MIMEBase subclass it is inserted directly into the resulting message attachments.

Parameters
- filename (*[type], optional*) – [description]. Defaults to None.
- content (*[type], optional*) – [description]. Defaults to None.
- mimetype (*[type], optional*) – [description]. Defaults to None.

attach_related_file(*path*, *mimetype=None*)

Attaches a file from the filesystem.

5.2 Synthea

apps.synthea.lib.utils.available_modules()

This method will load all the available modules that are in the folder *settings.SYNTHEA_MODULE_DIR*. Only files ending on .json will be loaded.

Returns Sorted list on module name with dicts holding the id and name of the module.
Return type List

apps.synthea.lib.utils.available_states()

This method will return a sorted list of available states based on the 'geography/demographics.csv' in the *settings.SYNTHEA_RESOURCE_DIR* folder.

Returns Sorted list on state name with dicts holding the id and name of the state.
Return type List

apps.synthea.lib.utils.run_synthea(*state*, *population=50*, *gender=None*, *age=None*, *module=None*)

This module will run the Synthea application on the background. This method expects Synthea to be installed on the *settings.SYNTHEA_BASE_DIR* location.

The output will be written to a unique folder in *settings.SYNTHEA_OUTPUT_DIR* that will be zipped and returned.

It will return the log and the zipfile location for futher processing. The zip file will not be deleted afterwards. So cleanup needs to be done manually.

Parameters
- state (*str, required*) – The state where to generate synthetic patient data for.
- population (*int, optional*) – The amount of patients to generate. Defaults to 50.
- gender (*str, optional*) – Either generate only male(m), only female(f), or None for both. Defaults to None.
- age (*str, optional*) – This is the age range of the generated patients. Input is always like [min_age]-[max_age]. Defaults to None.
- module (*str, optional*) – The module to use for generating patient data When None, all modules are used. Defaults to None.

Raises Exception – When the Synthea run fails it will return an Exception witht he Java error in it.
Returns The returning zipfile has the enabled options in the file name.
Return type (str,Path)

Chapter 6

Signals

6.1 API

apps.api.signals.create_user_token(*sender*, *instance=None*, *created=False*, ***kwargs*)

When a new user is created, this signal will also create a new API token for this user. So every user will have an API token.

Parameters

- `sender` (*sender*) – The model that has triggered the signal
- `instance` (`User`) – The newly created user model data
- `created` (*Boolean*) – Wether the object was created (True) or updated (False).

Chapter 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index

U